Amendments to the Specification

Please replace the paragraph on Page 10, lines 3 - 4 with the following marked-up replacement paragraph:

– Yet another object of the present invention is to provide this technique such that the [[queue]] queued updates are processed automatically when an event occurs. –

Please replace the paragraph that begins on Page 30, line 18 and carries over to Page 31, line 13 with the following marked-up replacement paragraph:

– A request to a WA object results in the back-end data source being updated in one of three possible modes: (1) a synchronous update; (2) an asynchronous update; or (3) a queued disconnected update, which may be referred to equivalently as a delayed update. Fig. 3B illustrates examples of the flows of this update process. As an application 335 executes, it requests some number of updates (shown as "351 ...") to a particular object, which in the example of Fig. 3B is HAO3 (element 360). If application 335 is operating in disconnected mode, then these updates cannot be processed against the back-end data source in real-time. Rather, the updates must be accumulated and applied to the back-end data source at some later time. In the branch office scenario, for example, the branch's daily work may be accumulated for transmission to the back-end enterprise system for processing after the branch closes for the day. Or, in the mobile computing scenario, the requests may be accumulated for subsequent transmission when the mobile device connects to a server. After transmission, the mobile device may then disconnect (to reduce connection costs, for example), and will receive the server's responses during some (arbitrarily-timed) subsequent connection. This queued disconnected mode is preferably used for the branch

office and mobile computing ~~scenario~~ scenarios, and may be used for other scenarios as well: application-specific criteria may be used to determine which update mode to use. --

Please replace the paragraph on Page 36, lines 3 - 15 with the following marked-up replacement paragraph:

-- When an object's script is re-executed by returning to Block 420 and an error condition persists, the processing preferably skips to the next element on the queue. This may be implemented by using a ~~retry-flag~~ retry flag or counter (not shown in Fig. 4). When the elements on a queue have been completely processed using the logic in Fig. 4 and this skipping of queued elements for repeating error conditions, occurrence of non-recoverable errors will result in a non-empty queue. In the preferred embodiment, this situation requires manual intervention to correct the associated problem. For example, an administrative interface may be provided to enable a systems administrator to address particular error conditions. Typically, non-recoverable errors will be system and application errors such as temporary or permanent unavailability of a back-end component. When errors are encountered that cannot be avoided at the present time, the updates remaining on the update queue may be suspended such that they will be scheduled for execution at a subsequent time, or they may be purged as part of the intervention process, as required by a particular implementation of the present invention. --